**Homework #7: Databases**

In this homework, you will be populating a database table with information about Pokemon (we have provided the cache data in pokemon.json) and writing code to fetch data from the table that you create.

Pokemon, short for ‘pocket monsters’ (TIL), is a video game series from Nintendo with a corresponding card game, television show, and several movies. Pokemon are creatures with special abilities that can be caught and taught to battle by Pokemon Trainers. Each pokemon has a unique level of stats that affect their fighting ability, including hit points (HP), speed, attack power, and defense power. Many Pokemon can evolve into new forms from gaining battle experience. Every pokemon has a type, some have more than one, that defines what type of abilities they can use (such as water, fire, or grass). Pokemon are most often caught by players to battle other pokemon, but other activities exist for players and pokemon to take part in such as fashion shows!

We have provided code for the following:

1. **read\_data\_from\_file():** To read the cache data in pokemon.json
2. **set\_up\_database():** To create an SQLite database and set up the ‘connection’ and ‘cursor’
3. **set\_up\_types\_table():** To create/setup one of the SQLite tables for you. This table is called ‘Types’. Run the starter code and then check the structure of the ‘Types’ table using the DB Browser!

**These functions are provided for you, DO NOT CHANGE THE CODE.**

When done with the assignment, your database will have two tables, including the one already provided for you. The other one, you will write the code for the create and fill!

We have also provided test cases that will pass if the functions are written correctly. Do **NOT** edit the test cases in any way.

Note: For extra credit, you will have to uncomment the **test\_get\_fastest\_pokemon\_of\_type()** function, and pass all the test cases within it.

**Tasks**

1. **create\_pokemon\_table(data, cur, conn):** This function takes 3 arguments as input: the data (written in JSON format), the database cursor, and database connection object. It returns nothing. It iterates through the data present in pokemon.json, creates a table Pokemon with the following columns:

- pokemon\_id (datatype: integer and Primary key)

- name (datatype: text)

- type1\_id (datatype: integer)

- type2\_id (datatype: integer or NULL)

- health\_points (datatype: integer)

- speed (datatype: integer)

- attack (datatype: integer)

- spl\_attack (datatype: integer)

- defense (datatype: integer)

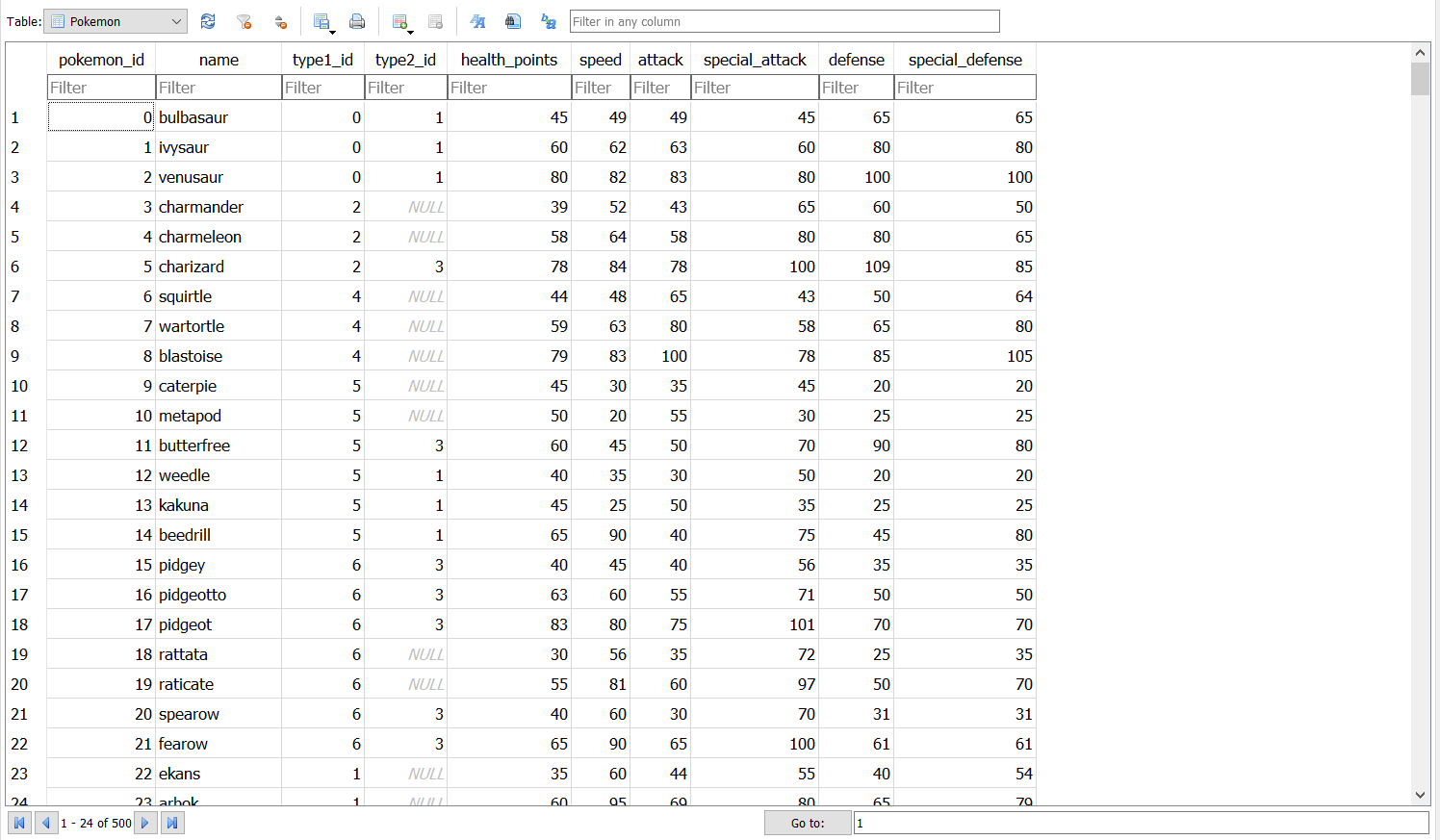
- spl\_defense (datatype: integer)

The function then loads all of the data present in pokemon.json into this table.

Note that some pokemon only have one type. In these cases, the value of type2\_id for that pokemon should be NULL in the database.

*Hint: To find the type1\_id and type2\_id values for each pokemon, you will first have to look up data in the Types table we create for you. See setUpTypesTable() and the ‘Types’ table itself in the DB Browser to get a better understanding.*

Expected table in DB Browser:



1. **get\_pokemon\_by\_attack\_range(attack\_min, attack\_max, cur):** This function takes three arguments as input: *attack\_min* value, *attack\_max* value, and the database cursor. It selects all pokemon which have an attack value inside that range (inclusive). It returns a list of tuples, each tuple containing a *pokemon\_id, name, and attack*.

For example, a pokemon named ‘test’, having an attack stat of 45 would be included in the returned values if the function was called as *get\_pokemon\_by\_attack\_range(40, 50, cur).*

**Example output for *get\_pokemon\_by\_attack\_range*(40,50, cur):**

**[(0, 'bulbasaur', 49), (6, 'squirtle', 48), (11, 'butterfree', 45),...(494, 'snivy', 45)]**

1. **get\_balanced\_pokemon\_above\_health(health\_min, cur):** This function takes two arguments as input: A minimum health value and the database cursor. It selects all "balanced" pokemon with a health level above the health\_min value. For the purpose of this function, a "balanced pokemon" is defined as a pokemon with a special attack value equal to its special defense value.

Note: pokemon that match the health\_min value should be included in the results

It returns a list of tuples, each tuple containing a *pokemon\_id, name, special\_attack, special\_defense, and health\_points*.  
  
**Example output for *get\_balanced\_pokemon\_above\_health(50, cur)*:**

**[(1, 'ivysaur', 80, 80, 60), (2, 'venusaur', 100, 100, 80), (10, 'metapod', 25, 25, 50),...(497, 'tepig', 45, 45, 65)]**

1. **get\_pokemon\_HP\_above\_speed\_attack(health\_points, speed, attack, cur):** The function takes five arguments as input: the HP, the speed, the attack, and the database cursor. It selects all the pokemon at an HP, speed greater than the speed passed to the function, with an attack greater than the attack passed into the function, and returns a list of tuples.

It returns a list of tuples, each tuple containing the pokemon *name*, *speed*, *attack*, and *defense*.

**Expected Output for pokemon with HP = 50, speed > 10 and attack > 60:**

**[('sandshrew', 40, 75, 85), ('bellsprout', 40, 75, 35), ('ponyta', 90, 85, 55),...('spiritomb', 35, 92, 108)]**

1. **get\_pokemon\_by\_type(type\_value, cur):** This function takes two arguments as input: a type value and the database cursor. It selects all pokemon which match that type and returns a list of tuples. Each tuple should contain a *pokemon\_id, name, and their type1 and type2 values*.

Note that the pokemon may match the type value as either their first or second type in the database. Additionally, the returned type1 and type2 values should be the text values of the types and not the id numbers.

*Hint: You have to use a JOIN for this task*.

**Example output for *get\_pokemon\_by\_type("grass", cur)*:**

**[(0, 'bulbasaur', 'grass', 'poison'), (1, 'ivysaur', 'grass', 'poison'), (2, 'venusaur', 'grass', 'poison'), … (496, 'serperior', 'grass', None)]**

**Grading Rubric**

1. **create\_pokemon\_table(): -** 15 Points
   1. 5 points for creating all 10 columns in the table (using the correct data type for each column)
   2. 5 points for entering all 500 pokemon in the table
   3. 5 points for properly using NULL when an entry does not have a second type value
2. **get\_pokemon\_by\_attack\_range() -** 10 points
   1. 5 points for correctly selecting the pokemon based on stated criteria
   2. 5 points for returning the list of tuples with the correct column values in each tuple
3. **get\_balanced\_pokemon\_above\_health() -** 10 points
   1. 5 points for correctly selecting the pokemon based on stated criteria
   2. 5 points for returning the list of tuples with the correct column values in each tuple
4. **get\_pokemon\_HP\_above\_speed\_attack() -** 10 points
   1. 5 points for returning a list of tuples of pokemon at the HP and above or equal to a speed and attack
   2. 5 points for returning the list of tuples with the correct column values in each tuple
5. **get\_pokemon\_by\_type() -** 15 points
   1. 5 points for correctly using a JOIN to get each pokemon’s type for their corresponding type\_id
   2. 5 points for correctly selecting the pokemon based on stated criteria
   3. 5 points for correctly returning the list of tuples with the correct column values in each tuple

**Extra Credit (6 points)**

**get\_fastest\_pokemon\_of\_type(type, cur):** This function takes two arguments as input: a type and the database cursor. It selects the pokemon of the specified *type* with the highest *speed* value. In the case of multiple pokemon with the same highest speed, all such pokemon should be returned.

It returns a list of tuples, with each tuple containing the pokemon name, type, and speed value.

**Example output for *get\_fastest\_pokemon\_of\_type*("ghost", cur):**

**[('froslass', 'ice', 110), ('gengar', 'ghost', 110)]**

**Make sure to uncomment the function test\_get\_fastest\_pokemon\_of\_type() for this portion of the homework. Your extra credit code must pass the commented test cases to receive credit.**